УДК 004.054

Животова А.А.,
Моденов Ю.Б., к.т.н., доц.

МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ WEB-ДОДАТКІВ

Національний авіаційний університет

oberst@nau.edu.ua

В останні десятиліття інформацію розглядають як один із основних ресурсів розвитку суспільства, а інформаційні системи та технології – як засіб підвищення продуктивності праці та ефективності роботи персоналу. Переробка інформації – найважливіша функція, без якої немислима цілеспрямована діяльність будь-якої системи

Ключові слова: системний тестовий сценарій, модель тестування за сценарієм, життєвий цикл, HTTP запит

Вступ

Програмування сьогодні перейшло з розряду мистецтва у клас ремесла для широкого кола фахівців. Дуже часто сучасні програмні продукти розробляються в стислі терміни і за умови обмеженого бюджету, що не сприяє підвищенню якості кінцевого продукту. На сьогодні тестування програмного забезпечення – один з найбільш дорогих етапів життєвого циклу програмного забезпечення, на нього відводиться від 50% до 65% загальних витрат. За умови вдалого застосування методів тестування можна домогтися високої ефективності процедури та вилучити максимальну користь.

Місце тестування у життєвому циклі програмного забезпечення

Основна концепція тестування – це базові терміни, ключові проблеми і їхній зв'язок з іншими областями знань. Тестування визначається як процес перевірки правильності програми в динаміці її виконання за тестовими даними. При тестуванні виявляються недоліки: відмови і дефекти як причини порушення роботи програми, збої як небажані ситуації, помилки як наслідки збоїв і ін. Базове поняття тестування – тест, що виконується в заданих умовах і за наборами даних. Тестування вважається успішним, якщо знайдено дефект або помилка, і вони відразу усуваються. Ступінь тестованості визначається критерієм покриття системи тестами, перевірки всіх можливих шляхів виконання програм і імовірності припущення стосовно того, що може з'явитися збій або помилкова ситуація в системі.

Перша створювана проміжна версія системи в інкрементальній моделі життєвого циклу реалізує частину вимог, в наступну версію додають додаткові вимоги і так до тих пір, поки не будуть остаточно виконані всі вимоги та вирішені завдання розробки системи. Для кожної проміжної версії на етапах життєвого циклу (ЖЦ) виконуються необхідні процеси, роботи та завдання, в тому числі, аналіз вимог та створення нової архітектури, які можуть бути виконані одночасно.

Згідно з цією моделлю ЖЦ, орієнтир робиться на розробку деякої закінченої проміжної версії, а завдання процесу розробки виконуються послідовно або частково паралельно для ряду окремих проміжних структур версії. Роботи та завдання процесу розробки наступної версії системи з додатковими вимогами або функціями можуть виконуватися неодноразово в тій же послідовності для всіх проміжних версій системи. Процеси супроводу та експлуатації можуть бути реалізовані паралельно з процесом розробки версії шляхом перевірки частково реалізованих вимог у кожній проміжній версії і так до отримання закінченого варіанту системи. Допоміжні та організаційні процеси ЖЦ зазвичай виконуються паралельно з процесом розробки версії системи і до кінця розробки будуть зібрані дані, на підставі

яких може бути встановлений рівень завершеності та якості виготовленої системи.

Особливість web-додатків як об'єктів тестування

Web-додаток – це система, яка зазвичай включає набір скриптів, розташованих на *web*-сервері і взаємодіючих з базами даних та іншими джерелами динамічного контенту. Вони швидко стають всюдисущими, оскільки дозволяють постачальникам послуг та їх клієнтам обмінюватися і маніпулювати інформацією (часто) незалежним від платформи способом за допомогою інфраструктури *Internet*.

Сама природа *web*-додатків – їх здатність порівнювати, обробляти і поширювати інформацію через *Internet* – робить їх двічі уразливими. По-перше, що саме очевидне, вони повністю відкриті з причини необхідності забезпечення публічного доступу. Це робить неможливим застосування техніки «*security through obscurity*» (безпеку за рахунок приховування інформації) і підвищує вимогу до стійкості коду. По-друге, що більш важливо з точки зору тестування на проникнення, вони обробляють дані, отримані із запитів *HTTP*-протоколу – протоколу, який може використовувати безліч способів кодування і інкапсуляції інформації.

Web-додатки, в першу чергу, характеризуються тим, що їх інтерфейс користувача має стандартизовану архітектуру, в якій :

- для взаємодії з користувачем використовується *Web*-браузер ;
- взаємодія з користувачем чітко розділяється на етапи, протягом яких браузер працює з одним описом інтерфейсу;
- ці етапи розділяються однозначно та виділяються зверненнями від браузера до додатка;
- для опису інтерфейсу застосовується стандартне уявлення (*HTML*);
- комунікації між браузером і *Web*-додатком здійснюються за стандартним протоколом (*HTTP*).

Тестувальник повинен використовувати всі доступні методи введення даних для досягнення виняткових умов всередині програми. Тобто вони не повинні бути обмежені тим, що надає оглядач чи автоматизовані засоби. *HTTP*-запити дуже легко генеруються за допомогою таких спеціальних програм, або скриптів оболонки. Процес вичерпного тестування за методом чорної скриньки включає в себе дослідження кожного елемента даних, визначення очікуваного введення, відтворення його, і аналіз результатів роботи програми на предмет виявлення ознак несподіваного поведінки.

Критичні зони web-додатків

Виходячи з вищевказаного, можна виділити такі основні критичні зони web-додатків, які потребують додаткового опису та інструкцій:

- єдність дизайну;
- навігація та «дружність»;
- функціональність;
- безпека;
- сумісність з оглядачами та операційними системами;
- працездатність.

Єдність дизайну *Web*-ресурсу є запорукою того, що ресурс справить враження цілісної структури, яка є гармонійною та логічною, тому така перевірка потребує чіткої документації, унікальної для кожного окремого ресурсу, проте не вимагає значних затрат ресурсів та часу.

Тестування продукту на навігацію та «дружність» допомагає спрямувати зусилля на покращення першого враження користувача про продукт та перетворити випадкову аудиторію на прихильників. Тому, цей аспект є доволі важливим, при цьому не потребує багато ресурсів, але бажана наявність чітких рекомендацій чи втручання спеціаліста з предметної області.

У переважній кількості розроблюваних додатків функціональність є ключовою зоною уваги як при тестуванні, так і під час користування кінцевим користувачем. Отже, такі перевірки вимагають ресурсів, досвіду від працівників, чітких вимог та рекомендацій з точки зору спеціалісту у відповідній сфері. Що є пози-

тивним моментом, то функціональні тестові сценарії добре підлягають автоматизації, що за необхідності частих перевірок скорочує час та ресурси.

Перевірка на безпеку є особливо критичним аспектом саме у тестуванні *web*-додатків, оскільки інтерфейс кожного користувача безпосередньо взаємодіє з єдиним сервером, та за певних умов та зусиль секретна інформація може опинитися в руках людини, що не повинна мати доступ для неї. Це може призвести до негативних наслідків та збитків як для представника послуг, так і для інших користувачів. Часто таке тестування може ефективно виконуватися командою з 1-3 людей з досвідом та кваліфікацією у відповідній сфері. Не залежить від специфіки певного *web*-ресурсу.

Сумісність з оглядачами та операційними системами є принципово важливою для продуктів, спрямованих на велику аудиторію (наприклад, інтернет-магазини, соціальні мережі). У час розвинених технологій та широкої різноманітності пристроїв, що мають доступ до мережі інтернет, кожен наявний пристрій додає перевірок. Звичайно, фізично неможливо, а за можливості - вкрай не рентабельно, проводити тестування на всіх існуючих пристроях, отже, враховуючи аудиторію користувача та популярність пристроїв, для перевірок обираються найбільш поширені з них. Таке тестування є ресурсозатратним незалежно від специфіки продукту.

Перевірка на працездатність є ключовою для ресурсів, що розробляються для великої кількості користувачів, що відвідуватимуть *web*-сервіс одночасно. Оскільки загалом такі тести виконуються за допомогою емуляторів та спеціальних програм - імітаторів навантаження, то така перевірка є автоматизованою, але вимагає високого рівня практики та знань від виконувача.

Місце системних сценаріїв у тестуванні web-додатків з метою оптимізації процесу тестування

Тестує інтегровану систему для перевірки відповідності всім вимогам. Крім того, системне тестування ПЗ повинно гарантувати, що програма працює так, як очікувалося, а також, що її не можна знищити або пошкодити її робоче середовище, яке викличе процеси в цьому середовищі, що переведуть систему в неробочий стан. Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог.

Мета системного тестування – визначення системи залежностей та підтвердження того, що необхідна інформація коректно та безпечно передається між компонентами системи. Системне тестування охоплює цілком всю систему. Більшість функціональних збоїв повинна бути ідентифікована ще на рівні модульних та інтеграційних тестів. У свою чергу, системне тестування, зазвичай фокусується на вимогах, що не відносять до функціональних – безпеки, продуктивності, точності, надійності т.п.

Дуже широко зустрічається практика автоматизації системних тестових сценаріїв для подальшого застосування такого тесту як тесту прийомки або санітарного тестування. До автоматизації є два підходи .

Перший підхід – це використовувати варіацію шаблону Модель-вигляд-контролер і формалізувати взаємодію користувача з інтерфейсом користувача в коді. Тоді системне тестування зводиться до тестування класів шаблону, а також логіки переходів. При такому підході з'являється проблема ініціалізації та приведення програми в потрібний для початку тестування стан .

Другий підхід – використовувати спеціальні інструменти для запису дій користувача. Тобто в підсумку запускається сама програма, але при натисканні на кнопки здійснюється автоматично. Сьогодні широко використовують *Selenium IDE*, який представлений як розширення

(плагін) до оглядача, або як середа розробки. Загальне правило автоматизації таке: на кожен варіант використання пишеться код, який описує дії користувача. Якщо всі варіанти використання покриті тестовими сценаріями і результати тестів відповідають очікуваним результатам, то можна вважати, що переважна більшість функціоналу реалізована коректно.

З підвищенням темпів росту сфери впливу *web*-додатків на життя пересічної людини, тестування програмного забезпечення є обов'язковим етапом в процесі розробки продукту, незалежно від обраної моделі життєвого циклу.

Системне тестування переслідує мету пошуку оптимізації підбору тестових сценаріїв таким чином, щоб тестове покриття було максимальним, як коли до циклу тестування включаються всі сценарії, та щоб час на перевірки був максимально скорочений, як коли до циклу включаються переважно високо пріоритетні сценарії.

Рис.1. Моделі тестування за сценаріями

Проектування системних сценаріїв є ефективним методом пошуку помилок для будь-якого типу програмного забезпечення чи додатку. Якщо при тестуванні програми, що встановлюється на комп'ютер та не потребує виходу в Інтернет це - можливість пошуку помилок алгоритму та програмної логіки, то для web-додатків це також і можливість виявлення проблем, скоєних збереженням чи не збереженням тимчасових даних, відсиланням завеликих чи зачастих запитів до сервера, розірванням з'єднання, навмисним наданням хибної інформації, що може призвести до помилок. Крім цього, підхід системних тестових сценаріїв допомагає у найкоротший шлях визначити критичні дефекти у декількох модулях програми за стислий час.

Висновки

В статті було розглянуто та проаналізовано місце тестування в процесі розробки програмного продукту та особливості тестування *Web*-додатків.

З використанням системних тестових сценаріїв можна домогтися підвищення ефективності процесу тестування. Цей метод є відомим, але не дуже поширеним. Сутність полягає у скороченні кроків для відтворення сценаріїв без втрати тестового покриття вимог. В результаті опрацювання тестового прикладу було створено 52 тестові сценарії (ТС) для покриття 8 функціональних вимог. Для відтворення всіх сценаріїв необхідно витратити 176 хвилин.

Після оптимізації сценаріїв було створено один сценарій з 58 простих кроків. Для відтворення системного сценарію необхідно витратити 14 хвилин.

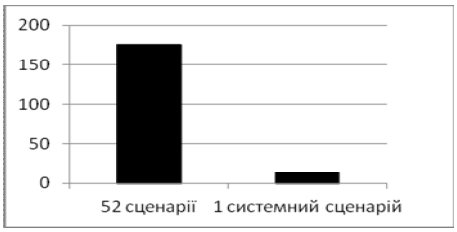

Рис.2. Діаграма порівняння часу, необхідного для тестування до та після оптимізації

Доведено, що пропонований метод скорочує час на тестування до 12,6 разів без втрати ефективності. Проте, не слід забувати також і про можливість комбінування такого підходу з традиційно вживаними, з метою запобігання втрати тестового покриття.

Список літератури

1. *Бейзер Б.* Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем - СПб. : Питер, 2004. - 320 с. -ISBN 5-94723-698-2.

2. *Калбертсон Роберт*, Браун Кріс, Кобб Гері Швидке тестування - М.: "Вільямс", 2002. - 374 с. - ISBN 5-8459-0336-X.

3. *Канер Сем*, Фолк Джек, Нгуен Енг Кек. Тестування програмного забезпечення. Фундаментальні концепції менеджменту бізнес-додатків - Київ: ДиаСофт, 2001. - 544 с. - ISBN 9667393879.

Статтю подано до редакції 19.05.2014